

# FUNCTIONS

---

Problem Solving with Computers-I

C++

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```



# Functions: Basic abstraction in programs

- Keep programs DRY !
- Three steps when using functions
  1. DECLARE: `void drawSquare(int y);`
  2. DEFINE: Write the actual code inside the function
  3. CALL: `drawSquare(20);`

You must always declare/define functions before calling them.  
Demo the use of functions

# Pass by value

```
#include <iostream>
using namespace std;

void bar(int x){
    x = x+5;
}

int main(){
    int y = 0;
    bar(y);
    cout<<y;
    return 0;
}
```

What is printed by this code?

- A. 0
- B. 5
- C. Something else

# While loops

A while loop is used to repeat code while some condition is true

```
while(BOOLEAN_EXPRESSION)
    //Code
}
```

Check if the `BOOLEAN_EXPRESSION` is true.

- \* If true, the statements in loop will execute.
  - \* at the end of the loop, go back to 1.
- \* If false, the statements in the loop will not execute.
  - \* the program execution after the loop continues.

# do-while loops

A while loop is used to repeat code until some condition is no longer true

```
do{  
    // Code  
    // This code is executed at least once  
}while(BOOLEAN_EXPRESSION);
```

1. Execute the code in the loop
2. Check if BOOLEAN\_EXPRESSION is true.
  - \* If true, then go back to 1.
  - \* If false, then exit the loop and resume program execution.

# C++ for loops

For loop is used to repeat code (usually a fixed number of times)

General syntax of a for loop:

```
for (INITIALIZATION; BOOLEAN_EXPRESSION; UPDATE) {  
    // code  
    // ...  
}
```

1. Execute the INITIALIZATION statement.
2. Check if BOOLEAN\_EXPRESSION is true.
  - \* if true, execute code in the loop.
    - \* execute UPDATE statement.
    - \* Go back to 2.
  - \* if false, do not execute code in the loop.
    - \* exit the loop and resume program execution.

# Continue and break

- `continue`;
  - can be used to stop the current iteration of a loop,
  - perform the UPDATE statement if necessary, re-check the `BOOLEAN_EXPRESSION`, and
  - continue with the next iteration of the loop.
- \* `break`; can be used to break out of the **current** loop and continue execution after the end of the loop.

```
for (int i = 0; i < 10; i++) {  
    if (i == 4)  
        continue;  
    if (i == 7)  
        break;  
    cout << "i = " << i << endl;  
}
```

# The accumulator pattern

Write a function that takes a parameter  $n$  and prints the sum of the series:

$$1 + 1/2 + 1/3 + \dots + 1/n$$

Write another function that returns the sum of the series



# Formatting output to terminal

See pages 91 and 190 of textbook

```
int i =10;
double j = 1/static_cast<double>(i);
cout.setf(ios::fixed);      // Using a fixed point representation
cout.setf(ios::showpoint); //Show the decimal point
cout.precision(3);
cout<<j;
```

What is printed by the above code?

- A. 0
- B. 0.1
- C. 0.10
- D. 0.100
- E. None of the above

# Nested for loops – ASCII art!

Write a function that prints a square of a given width

```
drawSquare( 5 );
```

```
* * * * *  
* * * * *  
* * * * *  
* * * * *  
* * * * *
```

# Draw a triangle

Which line of the drawSquare code  
(show on the right) would you modify  
to draw a right angled triangle

```
drawTriangle(5);
```

```
*
* *
* * *
* * * *
* * * * *
```

```
6   for(int i = 0; i < n; i++){ //A
7       for(int j=0; j < n; j++){ //B
8           cout<<"* "; //C
9       }
10      cout<<endl; //D
11  }
12      cout<<endl; //E
13
```

# Infinite loops

```
for(int y=0;y<10;y--)  
    cout<<"Print forever\n";
```

```
int y=0;  
for(;;y++)  
    cout<<"Print forever\n";
```

```
int y=0;  
for(;y<10;);  
    y++;
```

```
int y=0;  
while(y<10)  
    cout<<"Print forever\n";
```

```
int y=0;  
while(y=2)  
    y++;
```

# Next time

- Automating the compilation process with Makefiles
- Intro to lab02